C++

Samuele Amato

Contents

1 Storia di C++

I linguaggi di programmazione hanno subbito una grande evoluzione sin dal primo computer elettronico, all'inizio i programmatori lavoravano con quelle che erano le istruzioni più primitive: il **linguaggio macchina**, queste istruzioni erano formate da stringhe di 0 e 1.

Assembly presto diventa lo standard nel mondo della programmazione, rimpiazza (mappa) le ingombranti stringhe binarie con comandi leggibili dall'essere umano come **ADD** e **MOV**.

Essendo che i compiti eseguiti dai software sviluppati divennero sempre più complessi, i programmatori sentirono la necessità di un linguaggio capace di eseguire istruzioni matematiche relativamente complesse, che a loro volta erano combinazione di codici assembly, così nacque **FORTAN**, il primo linguaggio ad **alto livello** ottimizzato per calcoli numerici e scientifici, oltre ad introdurre concetti come sub-routine, funzioni e loop al mondo della programmazione.

Col tempo si sono sviluppati lingiaggi di livello superiore, come **COBOL** e **BASIC**, che hanno consentito ai programmatori di lavorare con qualcosa che si avvicina all'inglese, ad esempio `let i = 6`.

C stesso nasce come evoluzione rispetto alle sue versioni precedenti chiamate **B**, che era una versione migliorata di **BPCL**. Nonostante C fosse nato per aiutare i programmatori ad usare le funzionalità del nuovo hardware di quei tempi è diventato famoso grazie alla suaa velocità e portabilità. **C** è un linguaggio procedurale ed essendo che i computer si sono all'approccio orientato ad oggetti Bjarne Stroustrup inventò **C++**, che continua ad essere uno dei linguaggi di programmazione più utilizzati.

C++ ha implementato concetti della programmazione **object oriented** come l'**ereditarietà, incapsulamento, astrazione, polimorfismo**, termini che saranno spiegati più avanti, C++ continua ad essere utilizzato in molte applicazioni non solo perchè i nuovi linguaggi non soddisfano i requisiti di molte applicazioni, ma anchè per la sua flessibilità e potenza piazzata nelle mani del programmatore. C++ è regolarizzato dallo standard **ANSI** e continua ad evolversi.

2 Interpretatore e Compilatore

Un *interpretatore* traduce ed esegue un programma mentre lo legge, trasformando le istruzioni del programma o il codice sorgente, direttamente in azioni. Un *compilatore* traduce il codice sorgente in una forma intermediaria. Questa fase è chiamata **compilazione** e produce un file oggetto. Un applicazione di collegamento chiamata linker viene eseguita dopo il compilatore e combina il

file oggetto in un programma eseguibile contentente codice macchina che può essere eseguito direttamente sul processore.

Siccome gli interpreti leggono il codice così com'è scritto (leggono e eseguono il codice sorgente nel suo formato originale, senza la necessità di una fase di compilazione separata) ed eseguono immediatamente il codice, questi possono essere più facili da utilizzare per i programmatori. La maggior parte dei programmi interpretati è indicata come *script*, e l'interprete è indicato come *script engine*.

Il **compilatore** introduce lo step extra di compilare il codice sorgente (che è leggibile dagi umani in object code (che è leggibile dalle macchine). Questo step extra potrebbe sembrare non conveniente ma i programmi compilati vengono eseguiti molto velocemente essendo che la task di tradurre il codice sorgente in linguaggio macchina è già stata fatta, e non è richiesta quando viene eseuguito il programma.